

#2

31000 U.S. PTO
10/075642
02/13/02

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re the Application of : Noriyuki YOKOSHI
Filed: : Concurrently herewith
For: : APPARATUS AND METHOD FOR.....
Serial No. : Concurrently herewith

Assistant Commissioner for Patents
Washington, D.C. 20231

February 13, 2002

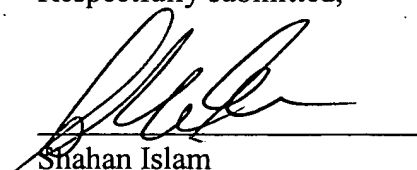
PRIORITY CLAIM AND SUBMISSION
OF PRIORITY DOCUMENT

S I R:

Applicant hereby claims priority under 35 USC 119 from **JAPANESE** patent application no. **2001-262358** filed **August 30, 2001**, a certified copy of which is enclosed.

Any fee, due as a result of this paper, not covered by an enclosed check, may be charged to Deposit Acct. No. 50-1290.

Respectfully submitted,


Shahan Islam
Reg. No. 32,507

ROSENMAN & COLIN, LLP
575 MADISON AVENUE
IP Department
NEW YORK, NEW YORK 10022-2584
DOCKET NO.: FUJI 19.449
TELEPHONE: (212) 940-8800

日 本 国 特 許 庁
JAPAN PATENT OFFICE

J1000 U.S. PRO
10/075642
02/13/02

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日

Date of Application:

2001年 8月30日

出 願 番 号

Application Number:

特願2001-262358

出 願 人

Applicant(s):

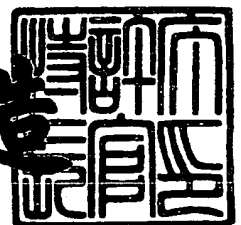
富士通株式会社

CERTIFIED COPY OF
PRIORITY DOCUMENT

2001年11月 2日

特 許 庁 長 官
Commissioner,
Japan Patent Office

及 川 耕 造



出証番号 出証特2001-3096109

【書類名】 特許願

【整理番号】 0150628

【提出日】 平成13年 8月30日

【あて先】 特許庁長官 及川 耕造 殿

【国際特許分類】 H04L 29/14

【発明の名称】 装置状態管理方法及びそのシステム

【請求項の数】 5

【発明者】

【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

【氏名】 余越 紀之

【特許出願人】

【識別番号】 000005223

【氏名又は名称】 富士通株式会社

【代理人】

【識別番号】 100070150

【住所又は居所】 東京都渋谷区恵比寿4丁目20番3号 恵比寿ガーデンプレイスタワー32階

【弁理士】

【氏名又は名称】 伊東 忠彦

【電話番号】 03-5424-2511

【手数料の表示】

【予納台帳番号】 002989

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9704678

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 装置状態管理方法及びそのシステム

【特許請求の範囲】

【請求項 1】 装置の状態を管理する機能であるMOをアプリケーションで実現し装置状態をデータベースに保持する装置状態管理方法において、

前記MOを前記データベース内部に実装し、

前記MOは、前記データベースから第1インタフェースを介して外部への制御を行うことを特徴とする装置状態管理方法。

【請求項 2】 装置の状態を管理する機能であるMOをアプリケーションで実現し装置状態をデータベースに保持する装置状態管理システムにおいて、

前記MOを前記データベース内部に実装し、

前記データベースから外部への制御を行う第1インタフェースを前記MOに実装したことを特徴とする装置状態管理システム。

【請求項 3】 請求項 2 記載の装置状態管理システムにおいて、

前記データベースから外部への制御に対する結果を前記制御に関連付けて前記データベースに通知する第2インタフェースを前記MOに実装してなることを特徴とする装置状態管理システム。

【請求項 4】 請求項 2 記載の装置状態管理システムにおいて、

前記データベースの外部の装置で変化した状態を前記MOに通知する第3インタフェースを前記MOに実装してなることを特徴とする装置状態管理システム。

【請求項 5】 請求項 2 乃至 4 のいずれか記載の装置状態管理システムにおいて、

前記データベースと前記外部の装置との間のプロトコルを変換するプロトコル変換部を有することを特徴とする装置状態管理システム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、装置状態管理方法及びそのシステムに関し、特に、装置の状態を管理する機能であるMOをアプリケーションで実現し装置状態をデータベースに保

持する装置状態管理方法及びそのシステムに関する。

【0002】

【従来の技術】

従来は、装置などの状態を管理する機能としてのMO (Managed Object) を一般的なアプリケーションとして実現し、ワークステーション (WS) やパーソナルコンピュータ (PC) 等の計算機内部で保持する必要がある情報について、データベースを利用して保持することで実現していた。この従来方法では、ユーザは、MOを管理する該当アプリケーションと、CMIPやSNMPなどのプロトコルを利用して制御や情報の取得を行う必要があった。

【0003】

一方、通信装置や計算機の状態を管理または制御する場合には、装置の状態だけの管理ではなく、例えば、利用者情報など管理対象と直接関連付けされない情報も必要で、上記利用者情報などを保持するデータベースも必須であった。

【0004】

図1は、従来方法を用いたネットワーク管理システムの一例のブロック図を示す。同図中、ネットワーク管理システム (NMS) 10は、GUI (Graphical User Interface) 等のユーザアプリケーション11と、WS, PC等の計算機12より構成され、伝送装置等の管理対象装置13を管理する。

【0005】

管理対象装置13の情報を扱うためには、計算機12の内部にMO15として情報を保持する必要がある。MO15内部の情報保管のためデータベース16, 17を利用している。情報を保持する必要性としては、第1に、NMS10側で管理対象装置13に設定した情報を保持しないで、管理対象装置13側だけに保持させると、管理対象装置13側の情報が消滅したときに、元の状態に戻すことが困難になる。

【0006】

第2に、NMS10側で設定した情報を読み出すとき常に管理対象装置13まで情報を取得にいく方法を採用すると、管理対象装置13とのインタフェースプ

トコルによって、NMS10からの制御・読出し時間が長くなるので、NMS10側に設定した情報を保持することで読出し時間を短縮できるので、図2に示すように、設定した情報をデータベース16に保持しておく。

【0007】

第3に、管理対象装置13内部の変化（障害情報、冗長構成部の切替え情報、運用状態など）情報をNMS10側に管理対象装置13から非同期通知できる場合には、図3に示すようにNMS10側でそのイベントを保持することで、常に管理対象装置13まで情報を取得にいかず、NMS10側で状態を確認することが可能となり、処理時間を短縮できる。上記の理由などで、管理対象装置13を監視制御するためのアプリケーションではモデル化によりMO15を定義し、NMS10側にも各種の情報をデータベース16などを利用して保持している。

【0008】

しかし、NMS10が扱う情報は全てMO15経由とは限らず、各管理対象装置13の設置場所等の一般的な情報はデータベース17の保持し、直接データベース17の情報も扱うことが多く、ユーザアプリケーション11は、NMS10に対し、MO15経由のインタフェースと、直接データベース17経由のインタフェースの2種類を扱う必要があった。

【0009】

【発明が解決しようとする課題】

従来においては、ユーザアプリケーションで扱う情報を保持・管理するために、一般的にはデータベースを利用するが、通常のデータベースでは、ユーザが格納した情報のみの管理であり、管理対象装置などの実際の情報もユーザが格納しない限り管理できない。ユーザの格納するタイミングによっては、実際の管理対象装置の状態と異なる情報になることもある。つまり、実体の管理対象装置の情報などは、一度データベースに格納しない限り、データベースのインタフェースでは、処理できない。

【0010】

また、通信機器や計算機の状態を管理するためには、実際の機器の状態を読み出したり、実際の機器に各種制御情報を設定したりする必要がある。このような

機能を実現するためには、MOとしてアプリケーションを実装することになるが、データベースとのインタフェース（SQL文によるアクセス）とは異なるSNMP、CMIP等のプロトコルをユーザアプリケーションは使用する必要があり、多くのアクセスインタフェースを実装しなければならない。つまり、データベースへのアクセスインタフェース以外に実体（管理対象装置など）にアクセスするためのインタフェースも必要となり、コスト高になる。

【0011】

更に、大規模なシステムを実現する場合には、処理の負荷分散を行うため、MOを実装するサーバとデータベースを実装するサーバを分散させて構築することが一般的である。しかし、この方法では、実際の情報を取得するために、ユーザアプリケーションから、MOを実装するサーバを経由してデータベースを実装するサーバへのアクセスとなり、直接データベースにアクセスするより処理時間がかかる。通信管理対象装置を監視制御するようなリアルタイム処理を要求されるシステムでは、処理時間を早くする必要があり、処理時間の遅れが問題となる。

【0012】

また、MOとデータベースが分離されて実装されている場合、情報の排他制御、ロック処理、操作のキャンセル、リカバリ等を行うトランザクション処理を実現するためには、データベースのトランザクション機能を利用することも可能であるが、MO側のアプリケーションでトランザクション機能を個別に実装する必要があり、アプリケーション作成のコストがかかるという問題があった。

【0013】

本発明は、上記の点に鑑みなされたものであり、ユーザアプリケーションからデータベースをアクセスすることでMO及び管理対象装置にアクセスすることができ、上記問題点を解決した装置状態管理方法及びそのシステムを提供することを目的とする。

【0014】

【課題を解決するための手段】

請求項1または2に記載の発明は、MOをデータベース内部に実装し、前記データベースから外部への制御を行う第1インタフェースを前記MOに実

装したことにより、ユーザアプリケーションは、データベースをアクセスすることでMO及び外部の装置を制御することができる。

【0015】

請求項3に記載の発明は、前記データベースから外部への制御に対する結果を前記制御に関連付けて前記データベースに通知する第2インタフェースを前記MOに実装してなることにより、

外部の装置の制御結果がMO及びデータベースに通知される。

【0016】

請求項4に記載の発明は、前記データベースの外部の装置で変化した状態を前記MOに通知する第3インタフェースを前記MOに実装してなることにより、

外部の装置からのイベントをMO及びデータベースで取り扱うことができる。

【0017】

請求項5に記載の発明は、前記データベースと前記外部の装置との間のプロトコルを変換するプロトコル変換部を有することにより、

データベースと外部の装置との間のプロトコルを変換して請求項2乃至4の発明を実現できる。

【0018】

付記6に記載の発明は、前記ユーザアプリケーションから前記データベースに対するトランザクション処理の完了前に、前記MOから外部への制御または外部から前記MOへの通知を行うことにより、

トランザクション処理の完了を待たずに高速の処理が可能となる。

【0019】

付記7に記載の発明は、前記データベースから前記ユーザアプリケーションに対しイベントを通知する第4インタフェースを前記MOに実装してなることにより、

データベースの内部変化をユーザアプリケーション側で確認でき、ユーザが所有している情報の同期をとることが可能となる。

【0020】

付記10に記載の発明は、前記第1インタフェースに、前記データベース上の

トランザクション処理を延長する機能を実装したことにより、

データベースのトランザクション機能で、MOにトランザクション機能を実現することができる。

【0021】

【発明の実施の形態】

本発明は、SNMP (Simple network management Protocol) や通信分野で一般的に利用されているMIB (Management Information Base) 情報で対象管理装置を表現したMOをいくつかの手段を実現することで、データベース内部に実装させる方法である。本発明ではデータベース内部にMOを実装させることで、一般的なデータベースと同じ扱いで、ユーザ側は、対象管理装置を表現したMOを管理することができる。

【0022】

図4は、本発明方法を用いたネットワーク管理システムの第1実施例のブロック図を示す。同図中、ネットワーク管理システム(NMS)20は、クライアントとしてのGUI等のユーザアプリケーション21と、NMSサーバ22より構成され、伝送装置等の管理対象装置13を管理する。そして、NMSサーバ22は、データベース24とプロトコル変換部25とよりなる。

【0023】

ユーザアプリケーション21は、管理対象装置13の状態を監視したり、管理対象装置13に各種パラメータを設定したりするために、データベース24内のMOを経由して情報の表示や制御をする。

【0024】

MOは、管理対象装置をモデル化して管理対象装置を機能単位に表現して、機能単位毎に情報と操作を持ち、データベース24内に実装されている。これにより、ユーザ側のユーザアプリケーション21からは、データベース24の操作だけでMOをアクセスできる。

【0025】

プロトコル変換部25は、ユーザアプリケーション21から指示されMO経由

で管理対象装置 13 を制御する場合に、NMS サーバ 22 内部の通信プロトコルを管理対象装置 13 との通信プロトコルに変換する機能である。なお、管理対象装置 13 との制御インタフェースは、管理対象装置 13 毎に固有のインタフェースが定義される。また、本発明では、管理対象装置 13 との通信プロトコルの変換機能だけでなく、データベース 24 内部に存在する MO とのインタフェースをする機能も含む。以下に主な機能を示す。

【0026】

1. データベース 24 内部の MO から制御メッセージを受け取る機能。

【0027】

2. 管理対象装置 13 へ制御をしたときの応答をデータベース 24 内の MO に返送する機能。

【0028】

3. 管理対象装置 13 から非同期に通知されるメッセージ（警報情報、管理対象装置内状態変化情報など）をデータベース 24 内の MO に通知する機能。

【0029】

図 5 は、データベース 24 内の MO の一実施例の構成図を示す。同図中、データベース 24 内には階層的に複数の MO 30, 31 が設けられている。例えば、MO 30 は伝送装置内の 1 つのユニットに対応するものであり、MO 30 はユニット内の 1 つのカードに対応するものである。

【0030】

各 MO 30, 31 には、少なくとも 1 つのテーブル 34 が設けられ、各 MO が保持する情報を格納する。MO の持つ情報によっては、複数のテーブル 32, 33 が設けられる。テーブル 32, 33, 34 それぞれには、テーブルを更新操作したときに実行されるメソッドや、テーブルにまとまった操作や複雑な操作を指示するためのメソッドを定義したストアードプロシジャ 36, 37, 38 が設けられている。以下に主なメソッドを示す。

【0031】

1. テーブルを更新する前、更新した後に自動的に実行されるメソッド。

【0032】

2. 直接、まとまった操作をテーブルに対して実行するメソッド。

【0033】

3. テーブル内の変化を外部に通知するメソッド。

【0034】

4. テーブル間でメッセージを通知するメソッド。

【0035】

本発明を実現するためのデータベース24と外部（ユーザアプリケーション21、プロトコル変換部25）のインタフェースや、データベース24内のインタフェースには、以下の種類のインタフェースが存在する。なお、丸付き数字は図5と対応している。

【0036】

① 一般的なテーブルをアクセスするための操作であり、質問用言語のSQL文を用いて、MOのテーブルを参照したり、更新したりする。

【0037】

② テーブルを更新（Insert, Updateなど）したときに自動的に実行されるインタフェースであり、必要に応じてストアドプロシジャで定義する。実行されるタイミングには、テーブルを更新する前と、更新後の2タイプが存在する。

【0038】

③ テーブルにまとまった操作を依頼したり、複雑な処理を依頼するためのインタフェースである。テーブル内に複数のレコードを指示した操作や、MO経由でデータベース外の管理対象装置に制御を依頼する場合に用いる。

【0039】

④ テーブル内に、変化が発生したときに、ユーザアプリケーションに通知するためのインタフェースである。例えば、管理対象装置からの警報情報をプロトコル変換部が認識し、データベースに情報を格納したときに、インタフェース②を利用してストアドプロシジャでユーザアプリに通知するような事象を実現する。

【0040】

⑤ ユーザアプリケーションからのインタフェース③や他のMOからの操作依頼（インタフェース⑨）などで管理対象装置への制御を依頼された場合に、操作情報を外部のプロトコル変換部に通知するために用いる。

【0041】

⑥ インタフェース⑤でプロトコル変換部に操作が依頼されたときの結果を送送するために用いる。

【0042】

⑦ 管理対象装置から非同期に通知される管理対象装置内の変化情報（警報情報や、状態変化通知）などを、データベース内のMOに格納するための操作である。直接、テーブルを更新するインタフェース以外に該当テーブルに存在するストアドプロシジャを利用するインタフェースも存在する。

【0043】

⑧ あるMOに対して操作が実施されたときに、他のMOに対して操作やテーブルの更新、参照を実行するために用いる。

【0044】

⑨ インタフェース⑧の逆方向の操作であり、他のMOからの操作やテーブル参照のインタフェースである。

【0045】

図5に示すインタフェース⑤～⑦を実装することで、外部の管理対象装置13への情報設定が可能となり、インタフェース⑤、⑥で外部の管理対象装置13の制御が可能となり、インタフェース⑦で管理対象装置13からの非同期通知情報のテーブルへの格納が可能となる。インタフェース⑤～⑦の実現方法は、一般的なデータベース製品によって異なるが、例として以下のようなメソッドを利用する。

【0046】

インタフェース⑤の実装方法としては、ユーザアプリケーション21からの操作の延長で、管理対象装置13への制御を実行することになるため、ユーザアプリケーション21から該当MOとのデータベースへのアクセスセッションと、プロトコル変換部25から同じMOに対するアクセスセッション間に通信をするた

めのルートを実現する。

【0047】

また、一部のデータベース製品では、パイプ機能を用いてプロセス間通信を実現することができる。但し、ユーザからの処理がトランザクション的に完了する前に、管理対象装置へのアクセスが必要となるため、データベースへのCommit（完了）処理以前に、処理が実行できる機能を用いる必要がある。

【0048】

インタフェース⑥は、インタフェース⑤によって操作がプロトコル変換部25に通知されたときに、その操作を実行した結果応答や、実行後に取得した情報をMO側に返送するために用いるインタフェースであるが、一般的なデータベースのストアドプロシジャで実現できる。なお、インタフェース⑥には、インタフェース⑤の制御と応答を対応付けるための情報が付与される。

【0049】

インタフェース⑦の実装方法については、管理対象装置13からの非同期通知メッセージ（警報情報や管理対象装置内の状態変化通知など）では、プロトコル変換部25がイベントを受信した後に、該当イベントを格納するMO側のテーブルにInsert（挿入）などのSQL文を用いる方法と、ストアドプロシジャで格納処理を記述する方法がある。

【0050】

また、本発明では、データベース24内にMO30、31を実装するため、ユーザアプリケーション21は、特殊なプロトコルを用いてMOをアクセスする必要がなく、データベース24をアクセスためのインタフェース①～④を実装するだけでよい。

【0051】

インタフェース①の実装方法としては、通常のSQL文によるSelect（参照）、Insert（追加）、Update（更新）操作を用いる。外部の管理対象装置13の情報は、インタフェース⑦により、MO30内部のテーブル32に格納されているため、ユーザアプリケーション21は、通常のテーブル上の情報として参照できる。

【0052】

また、管理対象装置13側の情報を変更したい場合には、テーブル32上の該当情報を更新したことによるトリガから、インタフェース②を利用してストアードプロシジャ36により、インタフェース⑤、⑥の処理を実行する方法と、直接インタフェース③を利用してストアードプロシジャ36を起動し、インタフェース⑤、⑥の処理を実行する方法とがある。

【0053】

インタフェース②の実装方法としては、データベース製品によって、実現方法は異なるが、1つの実装例としてデータベーストリガを利用する。このデータベーストリガとは、例えばテーブル32が更新（InsertまたはUpdate）されたときに、更新処理を実施する前か、実施後に処理の実行を宣言して自動的にストアードプロシジャ36を実行する機能である。

【0054】

インタフェース③は、複雑な処理または複数の処理を同時にMOに依頼するためのインタフェースであるが、一般的なストアードプロシジャを用いて実現する。

【0055】

インタフェース④は、MO内部で変化した状態（例えば、管理対象装置からの非同期通知イベントは追加された場合など）を外部のユーザアプリケーション21に通知するインタフェースであるが、実装方法にはいくつかの方法がある。

【0056】

a) ユーザアプリケーション21から、MO内部のテーブルの状態を周期的に読出し、変化を認識したとき情報を取得する方法であり、周期的にSelect（参照）を利用して変化を確認するか、周期的にストアードプロシジャを発行して変化情報を入手する。

【0057】

b) データベース製品によっては、データベース24内で発生した変化をALERT機能でユーザに通知することが可能なものもある。予めユーザ側で該当MOのテーブルにALERT通知の登録をすることで、非同期にデータベース内部で発生した変化を受信することができる。

【0058】

c) データベース製品によっては、データベースとの複数セッション間で通信ができる機能が実装されており、その機能を用いて、外部のユーザに通知することができる。

【0059】

本発明では、データベース24内にMO30, 31を実装するため、従来のようにMOとデータベースを個々に分散させて実装することはなく、データを共有することで、通信の情報量を削減でき、全体の処理時間を軽減できる。MO機能とデータベース機能を同じサーバに実装することによる負荷増加に対しては、複数プロセッサを実装して処理能力を上げることで回避できる。

【0060】

また、データベース24内にMO30, 31を実装することで、データベース24のトランザクション機能をそのまま利用してMO30, 31のトランザクション機能を実現することが可能となり、新たに開発する必要がなくなり、コストを低減できる。

【0061】

次に、具体的な実施例を示す。なお、操作文はデータベース製品により実装方法が異なることや、扱うプログラミング言語でも異なるため、論理的な記述として説明を行う。

【0062】

図6は、本発明方法を用いたネットワーク管理システムの第2実施例のブロック図を示す。同図中、図4と同一部分には同一符号を付す。図6において、ネットワーク管理システム(NMS)20は、クライアントとしてのGUI等のユーザアプリケーション21と、NMSサーバ22より構成され、伝送装置等の管理対象装置13を管理する。そして、NMSサーバ22は、データベース24とプロトコル変換部25とよりなる。プロトコル変換部25はネットワークエレメントマネージャである。

【0063】

この実施例では、管理対象装置13を管理するMOの一例として、データベー

ス24内に設けられたカードクラスMO40を例に取って説明する。1つの管理対象装置13には、複数のカード（パッケージ）が実装されており、カードを操作するための情報をカードクラスMO40内のカードテーブル41に格納する。また、カード内で発生したイベント情報（警報情報）を、カードクラスMO40内のイベントテーブル42に格納して管理する。また、カードテーブル41、イベントテーブル42それぞれにはストアードプロシジャ43、44が設けられている。

【0064】

図7（A）、（B）に、カードテーブル41、イベントテーブル42のスキーマ定義の一実施例を示す。図7（A）において、カラム名CARD_IDは複数パッケージを識別するプライマリーキーであり、CARD_NAMEは該当パッケージの名称情報（データベース24内だけに存在）であり、ALARM_INHは該当パッケージの監視をするかどうかのフラグであり、ALARM_STATEは該当パッケージの警報状態である。

【0065】

図7（B）において、カラム名EVENT_IDは複数パッケージを識別するプライマリーキーであり、EVENT_TIMEはイベントの発生時刻であり、CARD_IDは該当イベントの対応するCARD_ID情報であり、EVENT_MODEは障害情報の発生、回復情報であり、EVENT_STATUSは新規のイベントの状態を管理する情報である。

【0066】

カードテーブル41のストアードプロシジャ43について説明する。CARD_RESETは、該当カードに対してリセット操作を行うためのメソッドである。実際の管理対象装置13にだけ制御をするために、ストアードプロシジャで宣言する。

【0067】

インタフェース③としてのCARD_ALARM_MODEは、該当カードの障害情報を監視するかどうかを決める情報であり、実際の管理対象装置13に設定するための情報であることから、管理対象装置13への設定制御を実行したあ

とで、カードテーブル41上のALARM_MODEを変更するメソッドを定義する。

【0068】

インタフェース③としてのSET_RESPONSEは、CARD_RESETで操作した結果を外部のプロトコル変換部25から返送するためのメソッドである。

【0069】

イベントテーブル42のストアドプロシジャ44について説明する。インタフェース④としてのCHECK_EVENTは、新規にイベントが発生したことをチェックするためのメソッドである。

【0070】

次に、実際の操作例について説明する。まず、ユーザアプリケーション21からALM_INHモードを変更し、管理対象装置13まで制御をする場合について、図8に示す動作フローと共に説明する。図中の(イ)～(ヘ)は下記の説明と対応している。

【0071】

ユーザアプリケーション21の操作フローは、

(1) データベース22に対してストアドプロシジャCARD_ALARM_MODEを使用して制御をする(イ)。

【0072】

CARD_ALARM_MODE (1120, 1, 0, RSP)

TRX_ID = 1120 (トランザクションID)

CARD_ID = 1 (パッケージID)

INH_MODE = 0 (0: NOR, 1: INH)

(2) 応答結果のRSPを見て、0ならば正常終了 1または2ならば実行失敗である。

【0073】

データベース24内のストアドプロシジャ43の動作フローは、

(1) ユーザアプリケーション21から依頼されたCARD_ALARM_M

ODEのストアドプロシジャ43の中で、CARD_ID、INH_MODEをネットワークエレメントマネージャであるプロトコル変換部25へのメッセージに格納して管理対象装置13への設定依頼をする(ロ)。

【0074】

(2) 結果応答を別のメッセージパイプで待つ。

【0075】

(3) 正常ならば、データベース24を更新する。

【0076】

(4) 応答をユーザアプリケーション21に応答する(ヘ)。

【0077】

ネットワークエレメントマネージャであるプロトコル変換部25の動作フローは、

(1) MO40からの制御依頼メッセージを待つ。

【0078】

(2) 受信したメッセージから制御内容と、該当インスタンスを確認し、実際の管理対象装置13への制御を実行する(ハ)。

【0079】

(3) SET_RESPONSEを利用して管理対象装置13制御への結果情報をストアドプロシジャ43にて応答する(ニ)(ホ)。

【0080】

次に、ユーザアプリケーション21から管理対象装置13のカードにリセット制御を実施する場合について、図9に示す動作フローと共に説明する。図中の(イ)～(ヘ)は下記の説明と対応している。

【0081】

ユーザアプリケーション21の操作フローは、

(1) データベースに対してストアドプロシジャCARD_RESETを使用して制御をする(イ)。

【0082】

CARD_RESET (1130, 2, RSP)

TRX_ID = 1130 (トランザクションID)

CARD_ID = 2 (パッケージID)

(2) 応答結果のRSPを見て、0ならば正常終了 1または2ならば実行失敗である。

【0083】

データベース内のストアードプロシジャの動作フローは、

(1) アプリユーザから依頼されたCARD_RESETのストアードプロシジャの中で、CARD_IDをネットワークエレメントマネージャであるプロトコル変換部25へのメッセージに格納して管理対象装置13への設定依頼をする(ロ)。

【0084】

(2) 結果応答を別のメッセージパイプで待つ。

【0085】

(3) 応答をユーザアプリケーション21に応答する(ヘ)。

【0086】

ネットワークエレメントマネージャであるプロトコル変換部25の動作フローは、

(1) MO40からの制御依頼メッセージを待つ。

【0087】

(2) 受信したメッセージから制御内容と、該当インスタンスを確認し、実際の管理対象装置13への制御を実行する(ハ)。

【0088】

(3) SET_RESPONSEを利用して管理対象装置13制御への結果情報をストアードプロシジャ43にて応答する(ニ) (ホ)。

【0089】

次に、管理対象装置13からの警報情報をユーザアプリケーション21で認識する場合について、図10に示す動作フローと共に説明する。図中の(イ)～(ト)は下記の説明と対応している。

【0090】

ユーザアプリケーション21の操作フローは、

(1) ストアドプロシジャ44のCHECK_EVNTを利用して新規イベントを確認する。

【0091】

(2) もし、RSPが1ならば、EVENT_TBLに対して、EVENT_STATUSが1になっているレコードを読み出す(イ)(ロ)(ホ)(ヘ)(ト)。

【0092】

```
SELECT  EVENT_ID, EVENT_TIME, CARD_ID,
EVENT_MODE
FROM    EVENT_TBL
WHERE   EVENT_STATUS = 1;
```

(3) 受信したイベントのEVENT_IDをチェックし、最新のIDと最後のID間のEVENT_STATUSを0に更新する(読み取ったことをテーブルに更新する)。

【0093】

ネットワークエレメントマネージャであるプロトコル変換部25の動作フローは、

(1) 管理対象装置13から通知される非同期のメッセージをEVENT_TBLにINSERTする(ハ)(ニ)。

【0094】

```
INSERT INTO EVENT_TBL (EVENT_TIME, CARD_ID, EVENT_MODE)
VALUES (2000.09.30 13:00:15,
1, MAJ);
```

データベース24内のデータベーストリガによるストアドプロシジャ44の動作フローは、

(1) ネットワークエレメントマネージャであるプロトコル変換部25から新規のイベントが格納されたときに自動的に実行されるデータベーストリガにより

、EVENT__IDを自動的に付与する。

【0095】

(2) 新規のイベントであることを、EVENT__STATUSに付与する。

【0096】

(3) 同じデータベーストリガ内部で、CARD__IDと同じレコードのCARD__TBLのALARM__STATEを更新する。MAJならば1を設定し、CLRならば0を設定する。

【0097】

次に、ユーザアプリケーション21から管理対象装置13のカードに情報セット制御を実施する場合（正常終了時）の動作フローを図11に示す。

【0098】

ユーザアプリケーション21は、データベース24に対してストアードプロシジャCARD__SETを使用して制御し（イ）、データベース24内のストアードプロシジャはプロトコル変換部25に管理対象装置13への設定依頼をする（ロ）。プロトコル変換部25は、受信したメッセージから情報aを管理対象装置13に設定し（ハ）、管理対象装置13からの正常応答があると（ニ）、データベース24に設定情報aを更新保持する（ホ）。データベース24はユーザアプリケーション21に正常応答を返す（ヘ）。

【0099】

この正常応答を受けてユーザアプリケーション21はデータベース24に対してストアードプロシジャCARD__SETを使用して制御し（ト）、データベース24内のストアードプロシジャはプロトコル変換部25に管理対象装置13への設定依頼をする（チ）。プロトコル変換部25は、受信したメッセージから情報aを管理対象装置13に設定し（リ）、管理対象装置13からの正常応答があると（ヌ）、データベース24に設定情報aを更新保持する（ル）。データベース24はユーザアプリケーション21に正常応答を返し（ヲ）、ユーザアプリケーション21はデータベース24にコミットする（ワ）。

【0100】

このように、ユーザアプリケーション21は2つの処理を1つのトランザクシ

ョンとして扱い、2つの処理が正常応答になったときにデータベース24に対してコミットすることが可能となる。データベース24はコミットタイミングで情報aと情報bのデータを実際に更新する。なお、上記(ロ)(ハ)(ニ)(ホ)の動作、(チ)(リ)(ヌ)(ル)の動作は、トランザクションが完了していないときに外部への制御を行っている箇所である。

【0101】

図12は、ユーザアプリケーション21から管理対象装置13のカードに情報セット制御を実施する場合(異常終了時)の動作フローを示す。

【0102】

ユーザアプリケーション21は、データベース24に対してストアドプロシジャCARD_SETを使用して制御し(イ)、データベース24内のストアドプロシジャはプロトコル変換部25に管理対象装置13への設定依頼をする(ロ)。プロトコル変換部25は、受信したメッセージから情報aを管理対象装置13に設定し(ハ)、管理対象装置13からの正常応答があると(ニ)、データベース24に設定情報aを更新保持する(ホ)。データベース24はユーザアプリケーション21に正常応答を返す(ヘ)。

【0103】

この正常応答を受けてユーザアプリケーション21はデータベース24に対してストアドプロシジャCARD_SETを使用して制御し(ト)、データベース24内のストアドプロシジャは、プロトコル変換部25に管理対象装置13への設定依頼をする(チ)。プロトコル変換部25は受信したメッセージから情報aを管理対象装置13に設定し(リ)、管理対象装置13からの異常応答があると(ヌ)、データベース24に設定情報aを更新しない(ル)。データベース24はユーザアプリケーション21に異常終了応答を返し(ヲ)、ユーザアプリケーション21はデータベース24への処理をキャンセルする(ワ)。

【0104】

このように、ユーザアプリケーション21は2つの処理を1つのトランザクションとして扱い、2つ目の処理が異常終了応答であったためデータベース24に対してキャンセルを指示する。データベース24は一連の処理がキャンセルとな

ったため、データベース 2 4 への情報 a の更新処理も未実施で、元の情報となる。

【0105】

なお、上記の実施例は管理対象装置 1 3 に再操作が執拗ない場合を示しているが、管理対象装置 1 3 に設定した情報 a を元に戻す必要性がある場合には、異常終了応答（ヲ）の後に、再度ユーザアプリケーション 2 1 から情報 a を以前のの情報に設定する（イ）の制御を実施し、それに対する応答（ヘ）の後に、データベース 2 4 にキャンセルを指示する。

【0106】

図 1 3 は、複数ユーザアプリケーションを同一トランザクションとして動作させる処理を図 1 1 と同一シーケンスで記述した動作フローを示す。図 1 3 中の（イ）～（ワ）は図 1 1 と対応している。

【0107】

複数のユーザアプリケーション 2 1 a, 2 1 b が同じ管理対象装置 1 3 のカードに対して処理を実施する場合、データベース 2 4 上で同じリソースに対しての操作となるため、先に処理を実行したユーザアプリケーション側が処理を実施している間は、データベース 2 4 の機能によりロックがかかり、他のユーザアプリケーションは処理待ちになる。なお、本実施例では、同じ操作で説明するが、同じリソースに対する操作であれば、他の操作も処理待ちとなる。

【0108】

図 1 3 では、ユーザアプリケーション 2 1 b からの操作は、データベース 2 4 に依頼後に待ち状態となる。その後、ユーザアプリケーション 2 1 a の処理が完了後に処理が実行できる。つまり、データベース 2 4 内でユーザアプリケーション 2 1 b の処理が実行できるか確認し OK ならば可能である。しかし、例えばリソースの削除を行う処理が重なった場合は、後のユーザアプリケーションからの処理は、待ち合わせ後に該当リソースが存在しないことでデータベース 2 4 からエラー応答となる。つまり、トランザクションの種類によって、待ち合わせ後に並列処理となるか、エラー応答になるかである。

【0109】

次に、プロトコル変換部 25 について、更に詳しく説明する。プロトコル変換部 25 は、データベース 24 に対して管理対象装置 13 への処理依頼を定期的に確認し、その情報を実際の管理対象装置 13 に対して操作するフォーマット（プロトコル）に変換する処理部である。また、逆に管理対象装置 13 からの応答や非同期イベント通知を受信し、データベース 24 に格納する処理を実施する。

【0110】

プロトコル変換部 25 は、データベース 24 とはデータベース 24 の操作文法（質問用語）でアクセスを行い、管理対象装置 13 とは管理対象装置 13 に接続できるプロトコルで処理をする。例として管理対象装置 13 と CMIP プロトコルの場合を記述する。CMIP には、主な操作プロトコルとして以下のような操作が存在する。

【0111】

1. m-set 管理対象に情報を設定する処理。

【0112】

2. m-get 管理対象から情報を読み出す処理。

【0113】

3. m-action 管理対象に対して操作を実施する処理。

【0114】

4. m-create 管理対象に対し、リソースを生成指示する処理。

【0115】

5. m-delete 管理対象に対し、リソースを削除指示する処理。

その他、非同期通知側として

6. m-event-report 管理対象から非同期にイベント通知を受信する処理。

【0116】

上記 1～6 の処理にはいずれも、操作対象となる相手を識別するアドレス情報（DN 情報、実際にはクラス情報とインスタンス情報で構成される）や、動作情報（m-action の場合）や、属性識別子（m-get のときに、何の情報を読むか）や、属性値（m-set の場合にどの値を設定するか）などの情報が

付与される。

【0117】

プロトコル変換部25は、データベース24から指示があった情報を元に、管理対象装置13に対してアクセスをするために、管理対象装置13と通信ができるプロトコルに情報を変換して処理を実行する。また、管理対象装置13からの応答を該当プロトコルで待ち、結果をデータベース24の文法に変換してデータベース24に格納する。管理対象装置13とのプロトコル変換（実際に管理対象装置13へアクセスをする処理）は、ライブラリ等を利用することになる。

【0118】

例えば、CARD_ALARM_MODEの設定を行う場合、プロトコル変換部25では、管理対象装置13に対してALARM_MODEの状態を設定する処理となるため、データベース24から受信した情報（操作情報、管理対象アドレス情報、変更情報値）をm-setのコマンドにパラメータとして格納し、管理対象装置13に対して処理を実行する。データベース24からの情報は以下の通りである。

【0119】

CARD_ALARM_MODE

TRX_ID = 1120

CARD_ID = 1

INH_MODE = 0

管理対象装置13へのプロトコルは以下の通りである。

【0120】

m-set (cardクラス, cardアドレス情報: 1, 属性: INH_MODE, 属性値: 0)

また、CARD_RESETの処理を行う場合、プロトコル変換部25では、管理対象装置13に対してCARD_RESETの操作を実施するため、データベース24から受信した情報（操作情報、管理対象アドレス情報、変更情報値）をm-actionのコマンドにパラメータとして格納し、管理対象装置13に対して処理を実行する。データベース24からの情報は以下の通りである。

【0121】

CARD_RESET

TRX_ID = 1130

CARD_ID = 2

管理対象装置13へのプロトコルは以下の通りである。

【0122】

m-action (cardクラス, cardアドレス情報: 2)

なお、インタフェース⑤が請求項記載の第1インタフェースに対応し、インタフェース⑥が第2インタフェースに対応し、インタフェース⑦が第3インタフェースに対応し、インタフェース④が第4インタフェースに対応し、CHECK文がデータベースに対する質問用言語のMOへの操作を行う操作文に対応し、SELECT文がデータベースに対する質問用言語の参照文に対応する。

【0123】

(付記1) 装置の状態を管理する機能であるMOをアプリケーションで実現し装置状態をデータベースに保持する装置状態管理方法において、

前記MOを前記データベース内部に実装し、

前記MOは、前記データベースから第1インタフェースを介して外部への制御を行うことを特徴とする装置状態管理方法。

【0124】

(付記2) 装置の状態を管理する機能であるMOをアプリケーションで実現し装置状態をデータベースに保持する装置状態管理システムにおいて、

前記MOを前記データベース内部に実装し、

前記データベースから外部への制御を行う第1インタフェースを前記MOに実装したことを特徴とする装置状態管理システム。

【0125】

(付記3) 付記2記載の装置状態管理システムにおいて、

前記データベースから外部への制御に対する結果を前記制御に関連付けて前記データベースに通知する第2インタフェースを前記MOに実装してなることを特徴とする装置状態管理システム。

【 0 1 2 6 】

(付記 4) 付記 2 記載の装置状態管理システムにおいて、
前記データベースの外部の装置で変化した状態を前記MOに通知する第 3 インタフェースを前記MOに実装してなることを特徴とする装置状態管理システム。

【 0 1 2 7 】

(付記 5) 付記 2 乃至 4 のいずれか記載の装置状態管理システムにおいて
前記データベースと前記外部の装置との間のプロトコルを変換するプロトコル変換部を有することを特徴とする装置状態管理システム。

【 0 1 2 8 】

(付記 6) 付記 2 乃至 5 のいずれか記載の装置状態管理システムにおいて
前記ユーザアプリケーションから前記データベースに対するトランザクション処理の完了前に、前記MOから外部への制御または外部から前記MOへの通知を行うことを特徴とする装置状態管理システム。

【 0 1 2 9 】

(付記 7) 付記 2 乃至 6 のいずれか記載の装置状態管理システムにおいて
前記データベースから前記ユーザアプリケーションに対しイベントを通知する第 4 インタフェースを前記MOに実装してなることを特徴とする装置状態管理システム。

【 0 1 3 0 】

(付記 8) 付記 2 または 7 記載の装置状態管理システムにおいて、
前記ユーザアプリケーションから前記データベースに対する質問用言語に前記MOへの操作を行う操作文を設けたことを特徴とする装置状態管理システム。

【 0 1 3 1 】

(付記 9) 付記 2 または 7 記載の装置状態管理システムにおいて、
前記ユーザアプリケーションから前記データベースに対する質問用言語の参照文を用いて前記MOへの操作を行うことを特徴とする装置状態管理システム。

【 0 1 3 2 】

(付記 1 0) 付記 2 記載の装置状態管理システムにおいて、
前記第 1 インタフェースに、前記データベース上のトランザクション処理を延長する機能を実装したことを特徴とする装置状態管理システム。

【 0 1 3 3 】

(付記 1 1) 付記 2 記載の装置状態管理システムにおいて、
前記データベースとのセッション確立制御を拡張し、複数ユーザを同一トランザクションとして動作させることを特徴とする装置状態管理システム。

【 0 1 3 4 】

【発明の効果】

上述の如く、請求項 1 または 2 に記載の発明は、ユーザアプリケーションは、データベースをアクセスすることで MO 及び外部の装置を制御することができる。

【 0 1 3 5 】

請求項 3 に記載の発明は、外部の装置の制御結果が MO 及びデータベースに通知される。

【 0 1 3 6 】

請求項 4 に記載の発明は、外部の装置からのイベントを MO 及びデータベースで取り扱うことができる。

【 0 1 3 7 】

請求項 5 に記載の発明は、データベースと外部の装置との間のプロトコルを変換して請求項 2 乃至 4 の発明を実現できる。

【 0 1 3 8 】

付記 6 に記載の発明は、トランザクション処理の完了を待たずに高速の処理が可能となる。

【 0 1 3 9 】

付記 7 に記載の発明は、データベースの内部変化をユーザアプリケーション側で確認でき、ユーザが所有している情報の同期をとることが可能となる。

【 0 1 4 0 】

付記 1 0 に記載の発明は、データベースのトランザクション機能で、MO にトランザクション機能を実現することができる。

【図面の簡単な説明】

【図 1】

従来方法を用いたネットワーク管理システムの一例のブロック図である。

【図 2】

ユーザアプリケーションからデータベースへの情報設定ルートを示す図である

【図 3】

管理対象装置からデータベースへのイベント通知ルートを示す図である。

【図 4】

本発明方法を用いたネットワーク管理システムの第 1 実施例のブロック図である。

【図 5】

データベース内の MO の一実施例の構成図である。

【図 6】

本発明方法を用いたネットワーク管理システムの第 2 実施例のブロック図である。

【図 7】

カードテーブル、イベントテーブルのスキーマ定義の一実施例を示す図である

【図 8】

ユーザアプリケーションから ALM__INH モードを変更し、管理対象装置まで制御をする場合の動作フローである。

【図 9】

ユーザアプリケーションから管理対象装置のカードにリセット制御を実施する場合の動作フローである。

【図 1 0】

管理対象装置からの警報情報をユーザアプリケーションで認識する場合の動作

フローである。

【図 1 1】

ユーザアプリケーションから管理対象装置のカードに情報セット制御を実施する場合（正常終了時）の動作フローである。

【図 1 2】

ユーザアプリケーションから管理対象装置のカードに情報セット制御を実施する場合（異常終了時）の動作フローである。

【図 1 3】

複数ユーザアプリケーションを同一トランザクションとして動作させる処理の動作フローである。

【符号の説明】

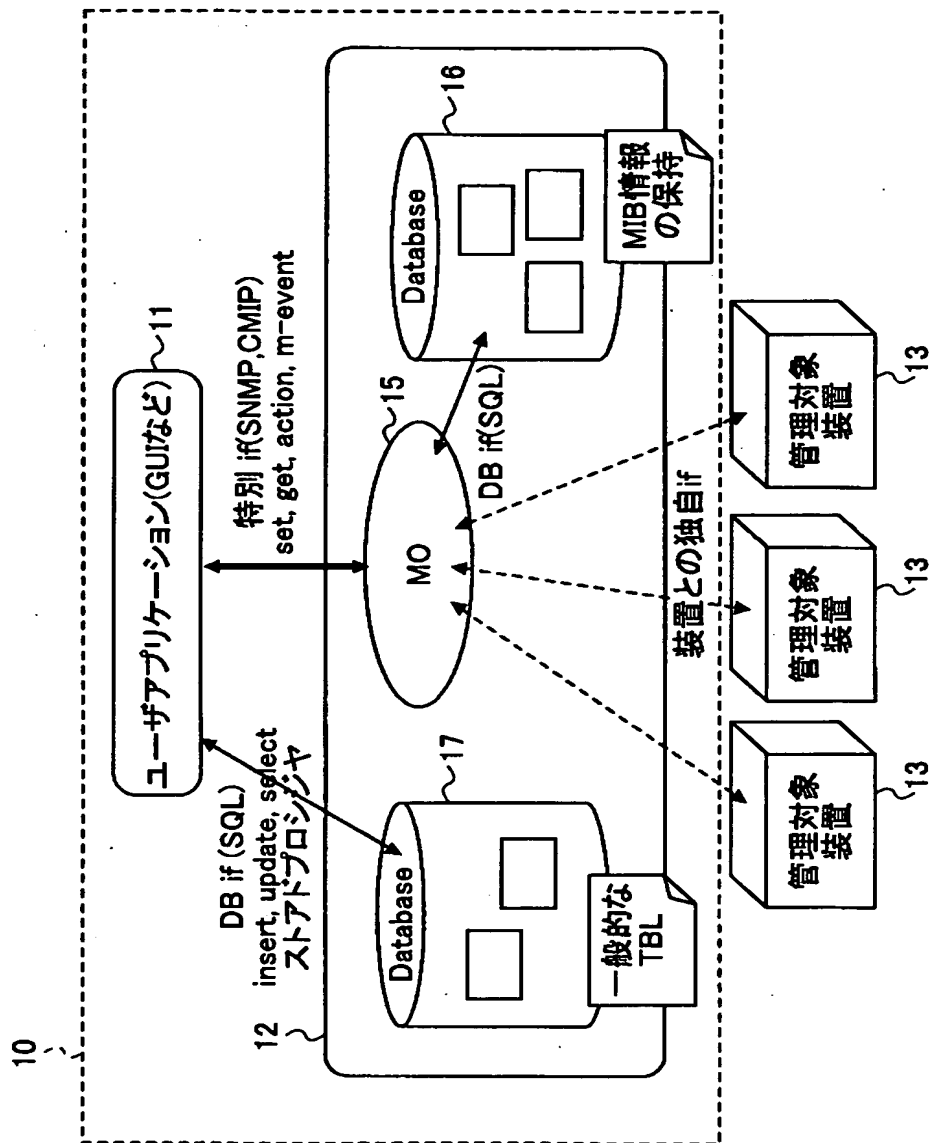
- 1 3 管理対象装置
- 2 0 ネットワーク管理システム
- 2 1 ユーザアプリケーション
- 2 2 NMSサーバ
- 2 4 データベース
- 2 5 プロトコル変換部
- 3 0, 3 1 MO
- 3 2, 3 3, 3 4 テーブル
- 3 6, 3 7, 3 8 ストアドプロシジャ

【書類名】

図面

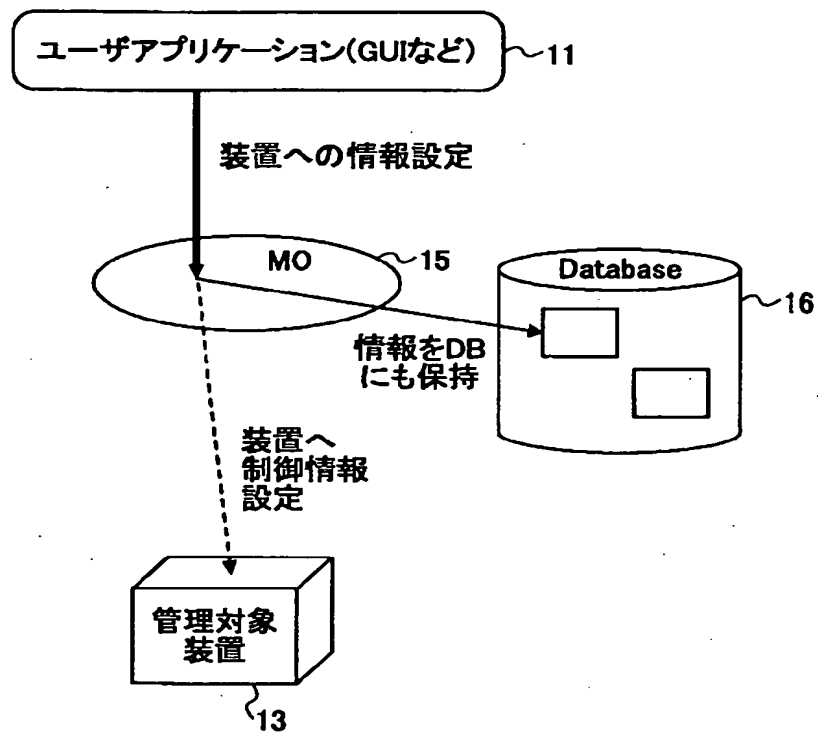
【図1】

従来方法を用いたネットワーク管理システムの一例のブロック図



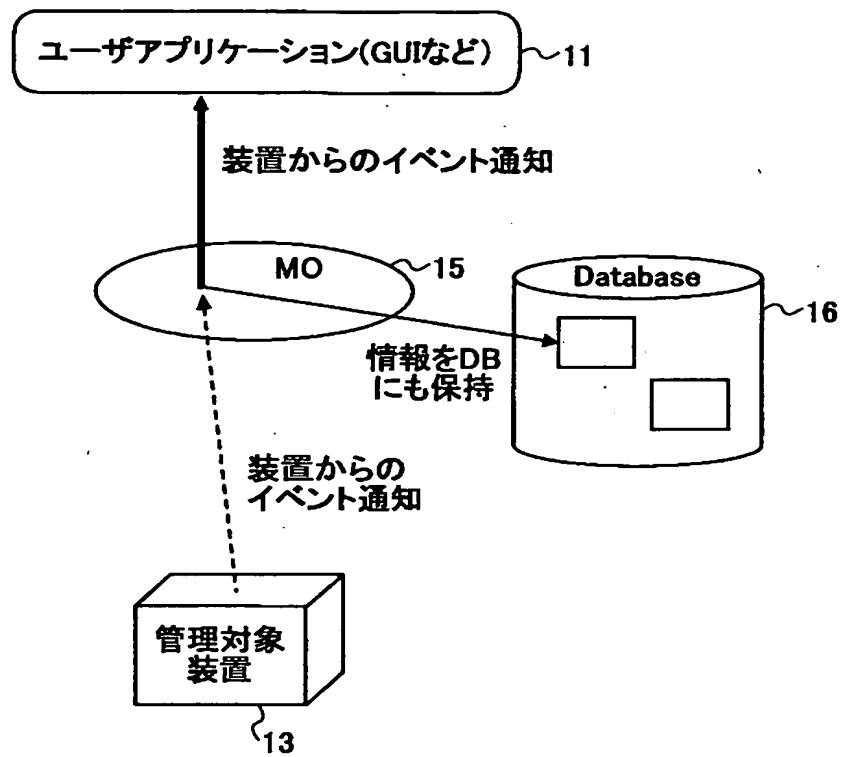
【図2】

ユーザアプリケーションからデータベースへの情報設定ルートを示す図



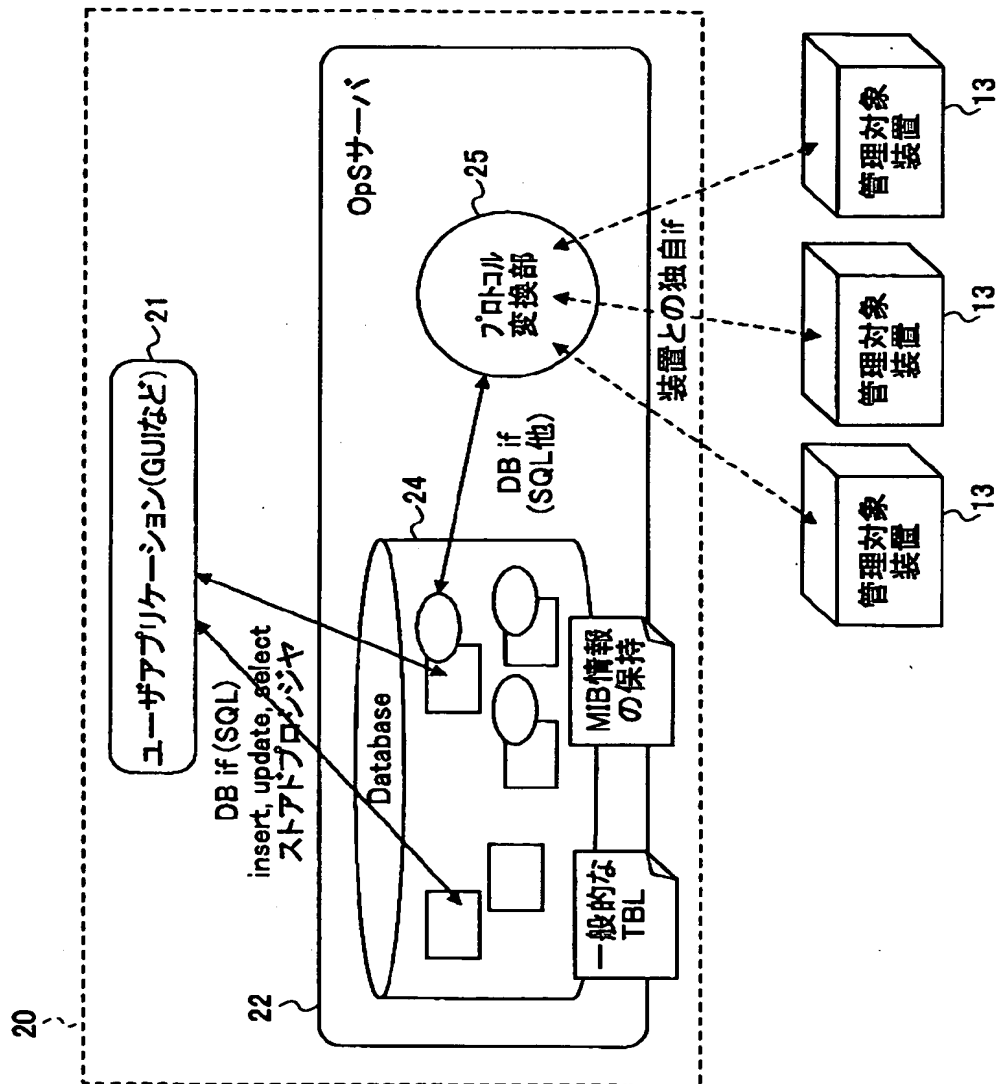
【図 3】

管理対象装置からデータベースへのイベント通知ルートを示す図



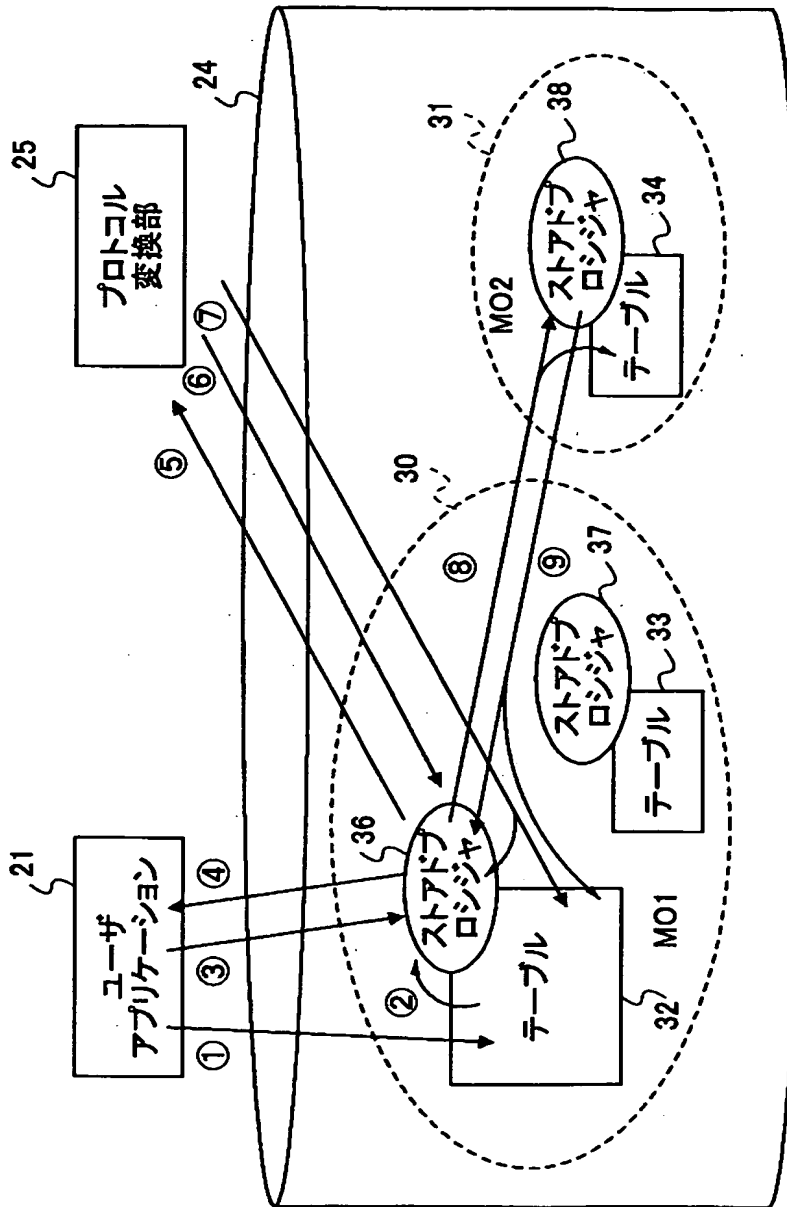
【図4】

本発明方法を用いたネットワーク管理システムの第1実施例のブロック図



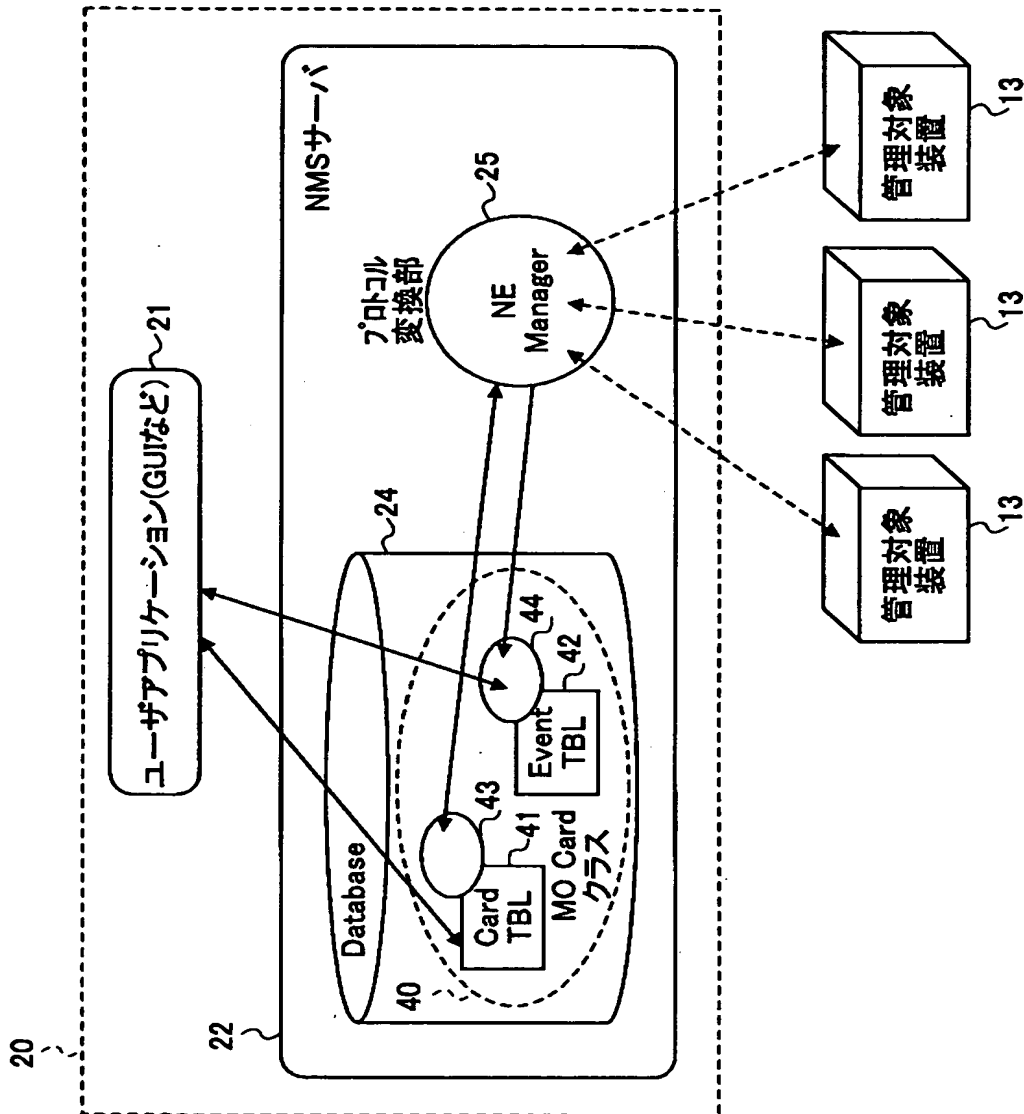
【図5】

データベース内のMOの一実施例の構成図



【図6】

本発明方法を用いたネットワーク管理システムの第2実施例のブロック図



【図 7】

カードテーブル、イベントテーブルのスキーマ定義の一実施例を示す図

(A)

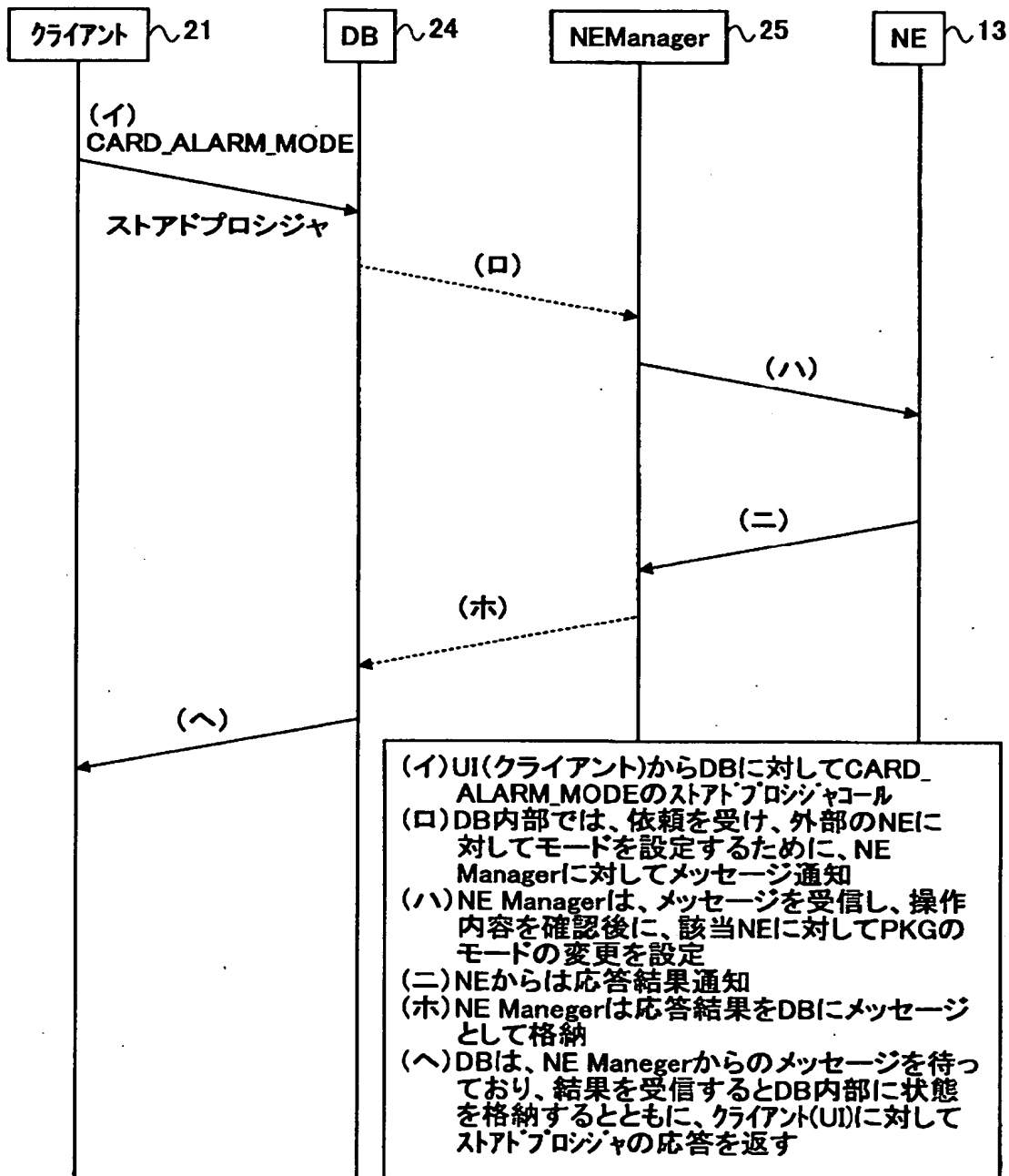
カラム名	型	備考
CARD_ID	NUMBER(10)	複数パッケージを識別するプライマリーキー
CARD_NAME	VARCHAR2(16)	該当パッケージの名称情報(DB内だけに存在)
ALARM_INH	NUMBER(2)	該当パッケージの監視をするかどうかのフラグ
ALARM_STATE	NUMBER(2)	該当パッケージの警報状態

(B)

カラム名	型	備考
EVENT_ID	NUMBER(10)	複数パッケージを識別するプライマリーキー
EVENT_TIME	DATE	イベントの発生時刻
CARD_ID	NUMBER(2)	該当イベントの対応するCARD_ID情報
EVENT_MODE	NUMBER(2)	障害情報の発生、回復情報
EVENT_STATUS	NUMBER(2)	新規のイベントの状態を管理する情報

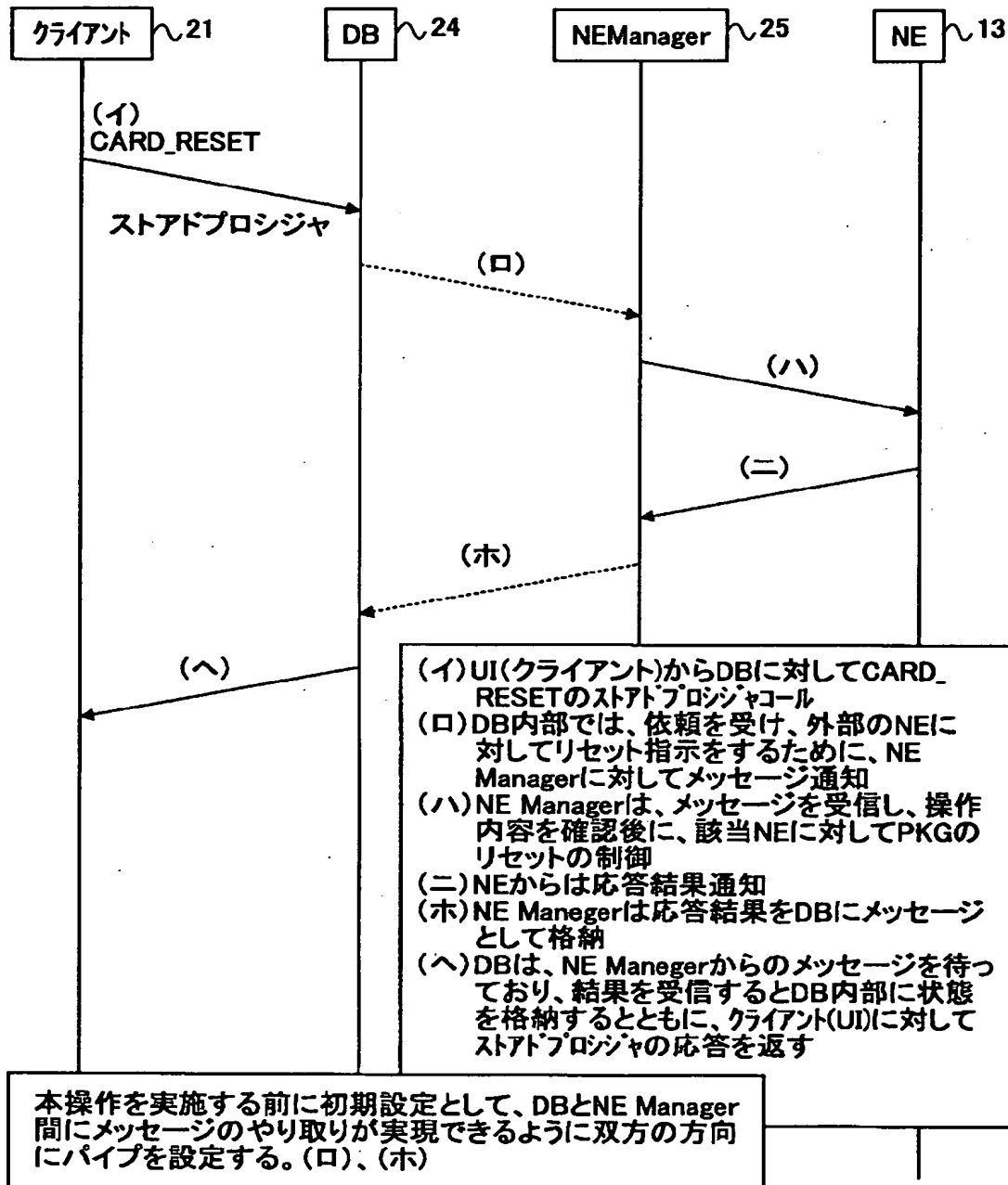
【図 8】

ユーザアプリケーションからALM_INHモードを変更し、
管理対象装置まで制御をする場合の動作フロー



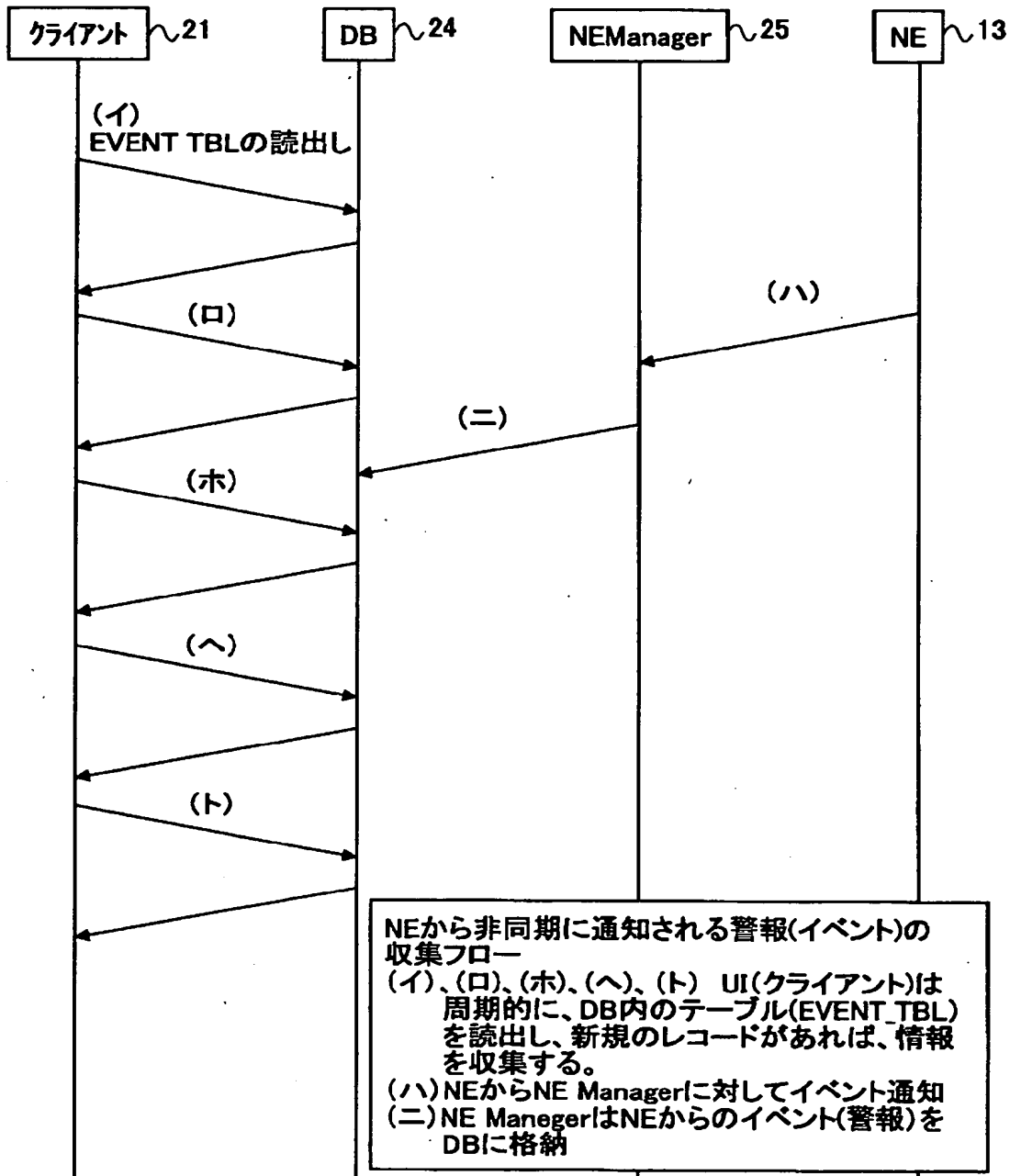
【図9】

ユーザアプリケーションから管理対象装置の
カードにリセット制御を実施する場合の動作フロー



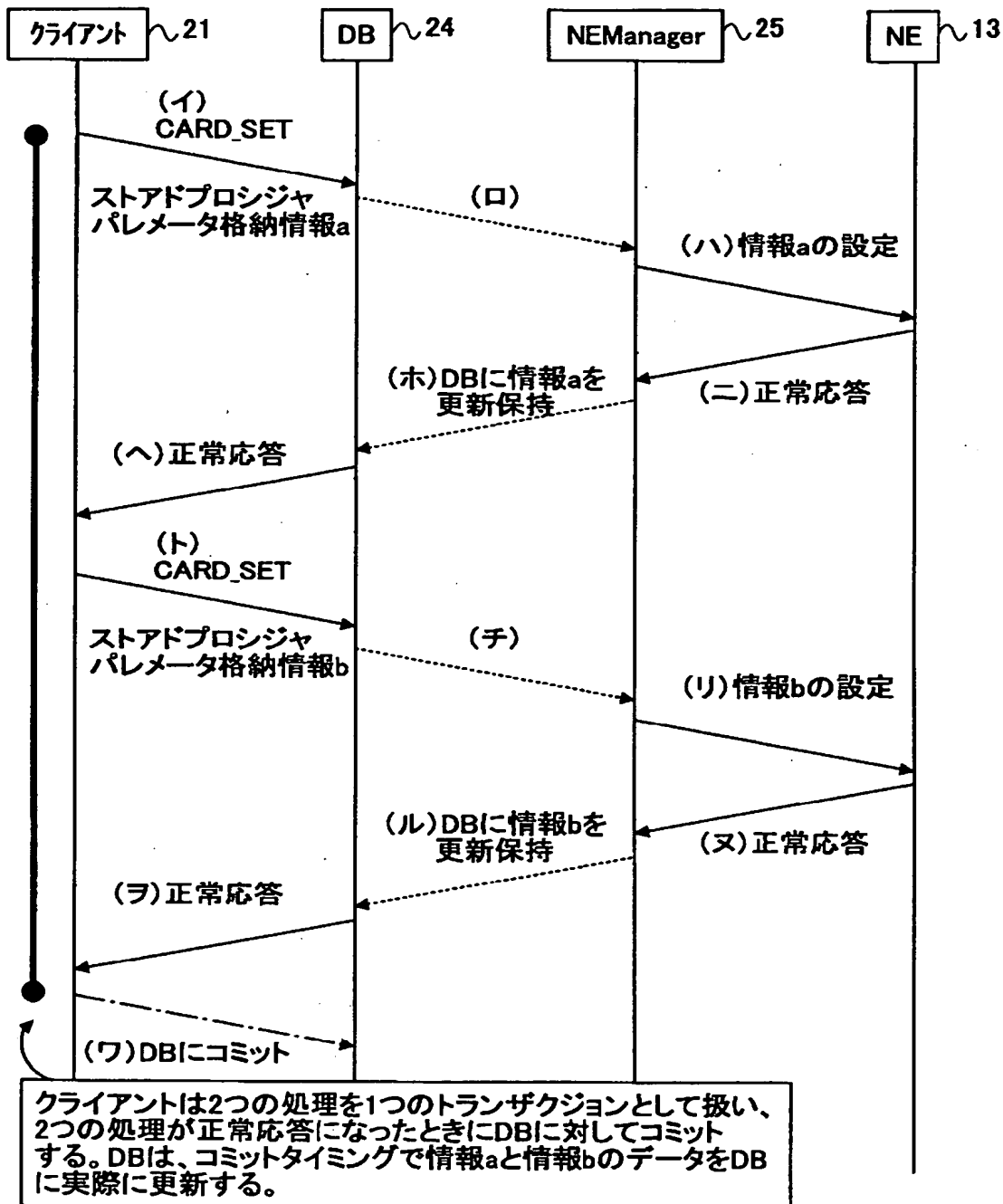
【図10】

管理対象装置からの警報情報を
ユーザアプリケーションで認識する場合の動作フロー



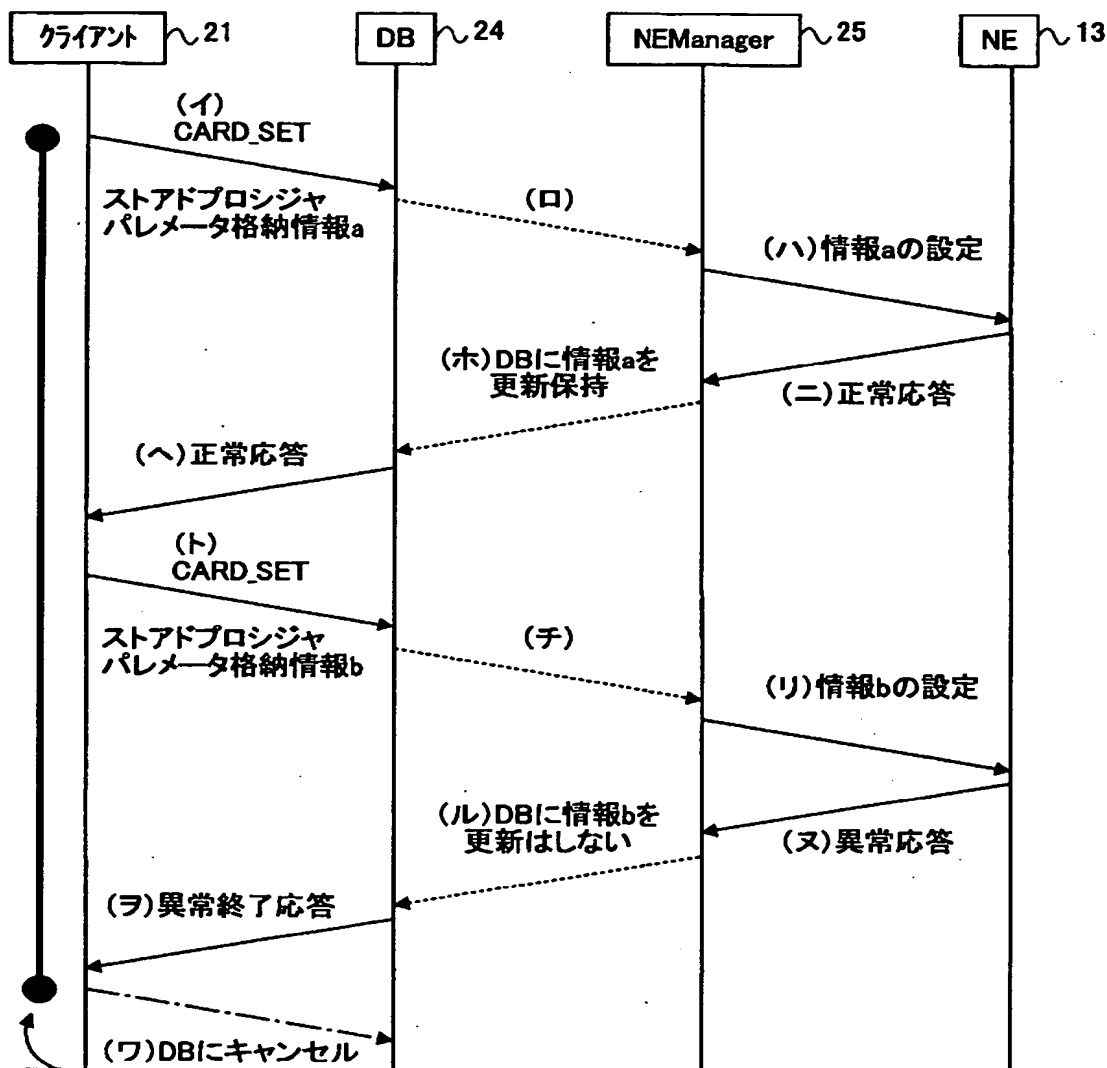
【図 11】

ユーザアプリケーションから管理対象装置のカードに
情報セット制御を実施する場合(正常終了時)の動作フロー



【図 1 2】

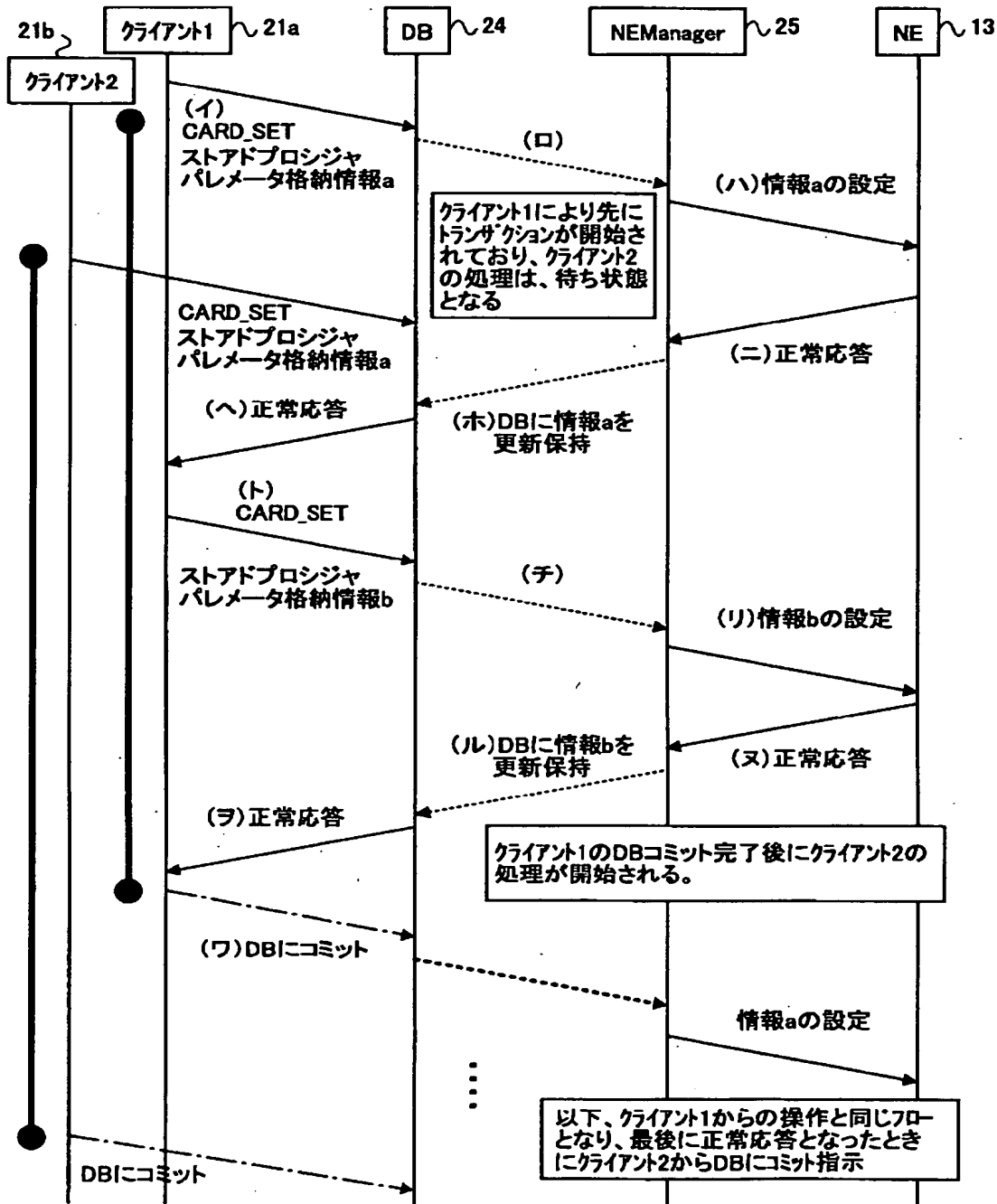
ユーザアプリケーションから管理対象装置のカードに
情報セット制御を実施する場合(異常終了時)の動作フロー



クライアントは2つの処理を1つのトランザクションとして扱い、2つめの処理が異常終了応答であったため、DBに対してキャンセルを指示する。DBは、一連の処理がキャンセルとなったため、DBへ情報aの更新処理も未実施で、元の情報となる。もし、NEの設定した情報aを元に戻す必要があるシナリオである場合は、(ヲ)の応答の後に、再度、クライアントから、(イ)の処理を以前の情報で再設定を実施し、その応答後にDBにキャンセルを指示する。
本シーケンスは、NEに再操作が必要ない場合を記述している。

【図 13】

複数ユーザアプリケーションを同一トランザクションとして動作させる処理の動作フロー



【書類名】 要約書

【要約】

【課題】 本発明は、ユーザアプリケーションからデータベースをアクセスすることでMO及び管理対象装置にアクセスすることができる装置状態管理方法及びそのシステムを提供することを目的とする。

【解決手段】 データベースから外部への制御の第1インタフェースをMOに実装し、MOをデータベース内部に実装し、ユーザアプリケーションは、MOをデータベースとして扱うことにより、ユーザアプリケーションは、データベースをアクセスすることで外部の装置を制御することができる。

【選択図】 図5

出 願 人 履 歴 情 報

識別番号 [000005223]

1. 変更年月日 1996年 3月26日

[変更理由] 住所変更

住 所 神奈川県川崎市中原区上小田中4丁目1番1号
氏 名 富士通株式会社